Ashdin Publishing
International Journal of Sensor Networks and Data Communications
Vol. 1 (2012), Article ID X110201, 13 pages
doi:10.4303/ijsndc/X110201

Research Article

Energy-Efficient Secure Routing Protocol Based on Roulette-Wheel and μTesla for Wireless Sensor Networks

Aly M. El-Semary[1,2]

[1] *College of Computer Science and Engineering, Taibah University, P.O. Box 30001, Al-Madina Al-Munawara, KSA*
[2] *System and Computer Engineering Department, Faculty of Engineering, Al-Azhar University, Cairo, Egypt*
Address correspondence to Aly M. El-Semary, aelsemary@taibahu.edu.sa

Received 15 February 2011; Revised 13 November 2011; Accepted 18 November 2011

Abstract Recently, wireless sensor networks (WSNs) have been deployed into a variety of applications including homeland security and military systems. Sensor nodes deployed in such networks are subject to several attacks including sinkhole, selective forwarding, wormhole, and spoofing attacks. Therefore, developing secure and energy-efficient routing protocols is imperative. This paper proposes an energy-efficient secure routing protocol for WSNs in which each sensor node forwards packets based on its own information. Thus it cannot be deceived by any other sensor node. The protocol employs the Roulette-Wheel selection algorithm to select a next node during the forwarding process while using μTesla protocol together with symmetric encryption and hash function algorithms to provide the needed security. Simulation results indicate that the performance of the proposed protocol outperforms the performance of the Path Energy Weight and the minimum hop protocols under such attacks. In addition, it exhibits a grateful performance under attack-free conditions.

Keywords WSN; wireless sensor network; energy-efficient; secure routing protocol; sinkhole attack; select forwarding; wormhole attack; Sybil attack

1 Introduction

Recently, wireless sensor networks (WSNs) have been emerged into a variety of applications including homeland security, military systems and health care. Each sensor node deployed in such networks has a limited energy and it is subject to several attacks such as sinkhole and select forwarding, wormhole and spoofing attacks. This requires the development of secure and energy-efficient routing protocols to protect the network against such attacks. Even though several routing protocols have been proposed in the literature, a number of these protocols including ones proposed by Heizelman et al. [8], Intagagonwiwat et al. [9], Chiang et al. [5], Azim [3], El-semary and Azim [7], and Lindsey and Raghayendra [12] concentrate on energy utilization of the deployed sensors while other protocols such as the ones presented by Deng et al. [6] and Zhou et al. [21] are designed to protect the network against a specific type attack. Furthermore, protocols such as the ones developed by Zhang et al. [20] and by Younis et al. [18] detect compromised nodes and provide an optimization strategy to avoid unnecessary overhead, respectively. Sensor nodes deploying any of these protocols forward packets based on information collected from other sensor nodes. This allows a malicious node to deceive the forwarding nodes to forward their packets through it and then enable several types of attacks such as sinkhole, select forwarding, wormhole and spoofing or Sybil attacks. Consequently, the significance of this paper is to propose an Energy-Efficient Secure Routing Protocol (EESRP) for WSNs that has the following merits: (1) evenly distributing the network load among the deployed sensor nodes and (2) protecting the associated wireless sensor network against such attacks.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work. Section 3 presents the proposed secure routing protocol. The experimental results are discussed in Section 4. Conclusions are presented in Section 5.

2 Related work

Due to the natural constraints imposed on sensor nodes, several network-layer protocols have been proposed to utilize sensor's energy to prolong the lifetime of deployed wireless sensor networks (WSNs). According to Akkaya and Younis [1], these protocols can be generally classified into three categories: data-centric, hierarchal and location-based protocols. Extensive surveys on these routing protocols are presented by Akkaya and Younis [1] and Al-Karaki and Kamak [2].

Data-centric protocols are negotiation-based and application-specific protocols. Several data-centric protocols have been proposed including Directed Diffusion proposed by Intagagonwiwat et al. [9]. In Directed

Diffusion, data generated by sensor nodes is named by attribute-value pairs. When a node requests data, it sends an interest that specifies this data. Data matching the interest is then "drawn" into the originators of the interest through multiple paths. The Directed Diffusion protocol achieves the energy saving by reinforcing one of these paths and by caching and processing in-network data.

Hierarchal routing protocols often group sensor nodes into clusters that form a hierarchy. Several hierarchal protocols have been introduced including LEACH introduced by Heizelman [8] and PEGASIS presented by Lindsey and Raghayendra [12]. LEACH stands for Lower-Energy Adaptive Clustering Hierarchy. It is a clustering protocol that randomly rotates the local cluster heads to balance the energy load among the deployed sensors in the network. The main goal of LEACH is how to choose the cluster head which in turn receives data from other nodes in its cluster, makes data aggregation, and then sends it to the base station. PEGASIS (Power-Efficient Gathering in Sensor Information System) is a chain-based algorithm to eliminate the overhead in dynamic formation of cluster. It uses greedy algorithm to data chain. Each node aggregates data from downstream node and sends it to upstream node along the chain.

Location-based routing protocols can be further classified into (1) routing protocols deploying only location information of nodes such as GEAR proposed by Yu et al. [19] and (2) routing protocols that use both location and energy information of sensor nodes. Routing protocols that depend on the location and energy include Min-Hop proposed by Chiang et al. [5], MAP presented by Azim [3], and PEW introduced by El-semary and Azim [7]. The Min-Hop (minimum hop) chooses an optimal path to the sink node by forwarding data packets into the node that has the short distance to the sink. The distance is represented by a number of hops to the sink. Thus, the node that has the minimum hop count is the one that has the shortest distance. If several nodes have the same distance, the one with the maximum energy is considered. Although the Min-Hop improves the average energy consumption in the network, it over utilizes the nodes along the shortest paths. This results in increasing energy gap and decreasing network lifetime. MAP is an enhanced version of the Min-Hop to lengthen the network lifetime. The MAP distributes network load by choosing the neighbor node with the maximum energy but this leads to significantly longer paths. Both Min-Hop and MAP forward data packets based on local view of neighbors and thus may result in energy holes that lead to decreasing the network lifetime. In other words, selected paths may include one or more nodes with a very low energy. PEW (Path Energy Weight) overcomes this shortcoming by deploying a global view mechanism. Its main idea is to map energy levels of all nodes along the communication path into a single parameter that gives an estimation of how uniform is the energy distribution along the path by providing better weight to a path with balanced energy level over a path with unbalanced energy level.

Although the above-mentioned routing protocols utilize the limited capabilities of sensor nodes, they have not been designed with a security goal in mind. Consequently, they are not applicable into adversarial environment, such as military systems and disaster relief, due to their susceptibility to a great number of attacks. Karlof and Wagner [11] introduced these attacks, analyzed their applicability and provided numerous solutions. These attacks include selective forwarding, sinkhole, wormhole and spoofing attacks.

In selective forwarding attack, malicious nodes may not forward specific messages. A simple aspect of this attack is when a malicious node acts like a black hole by refusing to forward every observable packet. This attack is most effective when the attacker's node is able to include itself on the path of target messages. The sinkhole attack prevents the base station from obtaining complete and correct sensing data, thus forming a serious threat to higher-layer applications. The sinkhole attack is achieved by making a compromised node look attractive to its neighbor nodes with respect to the routing metrics. Consequently, the attacker manages to draw as much traffic as possible that is designated to the base station. By involving itself in the routing process, it is then able to launch more sever attacks such as selective forwarding, modifying or dropping the received packets. Wormhole attack forms a severe threat against packet routing in WSNs. In this attack, an adversary receives packets at one location in the network and tunnels all or part of these packets to another location in the network, where the packets are sent back into the network. In a spoofing or Sybil attack, an attacker can easily inject bogus packets by impersonating another sender. The attacker can also easily eavesdrop on communication, record packets, and then reply the packets.

To protect WSNs against the associated attacks, a number of secure routing protocols have been proposed. These include SPINS [16], ITSRP [21], ASP [18], COOL [20], SeMuRa [4] and the one proposed by Kanjee et al. [10].

SPINS enabled security features through two security protocols: SNEP and μTesla. SNEP provides confidentiality, integrity, authentication and freshness, while μTesla [14, 15] provides authenticated broadcast. SPINS gives more attention on key management. ITSRP is a secure routing protocol proposed by Zhou et al. They focused on exchanging session keys that are used to encrypt messages. If a source node wants to send a secret message to the sink node, it must first exchange a session key which in turn will be used to encrypt the message. ASP (Adaptive Security Provision) is a security model proposed by Younis et al. [18]. Their model avoids unnecessary overhead imposed on individual nodes. Their idea is to adapt the security provision to the application needs and the trust level of participated nodes. COOL (COmpromised nOde Locator) is an authentication

scheme proposed by Zhang et al. The main idea of this protocol is to detect and locate compromised nodes once they misbehave in the network based on the observation of well-behaved sensor nodes. SeMuRa proposed by Triki et al. [4] presented a novel secure and multipath routing algorithm in wireless sensors networks. Their algorithm tolerates node failures and improves the reliability of data routing. Kanjee et al. [10] proposed a two-tier authentication scheme based on physiology. The scheme lies on their unique secure architecture for Healthcare WSN that explores the characteristics of Healthcare WSN.

Finally, a protocol called μTesla [15] has been proposed for broadcast authentication in distributed sensor networks. Generally, the broadcast authentication is implemented using asymmetric mechanisms but due to the high communication, computation and storage overheads of the asymmetric cryptographic mechanisms, it is impractical to implement them in resource constrained sensor networks. Therefore, μTesla introduced asymmetry by delaying the disclosure of symmetric keys. In this protocol, the network lifetime is divided into n time intervals and a chain of authentication keys $K = \{k_0, k_1, \ldots, k_n\}$ is generated. The keys in K are linked to each other by a one-way function and they are obtained by first choosing a random value k_n as the last key in K and then continuously executing a one-way function f to compute all the other keys: $k_i = f(k_i + 1)$, $0 \leq i \leq n-1$. Each k_i is assigned to authenticate all broadcast messages sent in the ith time interval, $1 \leq i \leq n$, and k_0 is the initial key which refers to the commitment of K. If a key k_j is given, only the previous keys k_i can be computed using f, $0 \leq i \leq j-1$, but the later keys k_i cannot be computed, $j+1 \leq i \leq n$. Therefore, with the knowledge of k_0, any other key in K can be authenticated by just performing f. When a sender wants to send a message in ith time interval, the sender generates a Message Authentication Code (MAC) for this message with a key derived from k_i and then broadcast both the message and its MAC. The key k_i will be disclosed after a certain period of time which is called the disclosure lag. After receiving the packet that includes the message and its MAC, the receiver first ensures that the packet is received before the key was disclosed. If so, the packet is buffered until receiving its corresponding key. Otherwise, the packet is ignored.

3 Proposed Energy-Efficient Secure Routing Protocol

This paper introduces an Energy-Efficient Secure Routing Protocol (EESRP) that provides security for data packets during its way from a source node to the sink node and also prolongs the network lifetime by evenly distributing network load among deployed sensors. The EESRP is achieved by employing two novel protocols: Roulette-Wheel Routing Protocol (RWRP) and Secure Routing Protocol (SRP). The RWRP discussed in Section 3.2 is concerned with routing data packets while SRP introduced in Section 3.3 is interested in securing these packets during their traveling across the network. The security aspects of EESRP protocol are presented in Section 3.4. Before demonstrating the proposed protocol details, the paper first adopts the protocol assumptions and notations. In addition, it presents the network model in Section 3.1.

Protocol assumptions.
The proposed protocol adopts three assumptions: (1) secret keys should be kept secret among the intended sensor nodes, (2) secret keys shared among authorized nodes should not be revealed to others and (3) each authentic sensor node should follow the protocol steps.

Basic notations used in the proposed protocol
$\mathrm{E}(m,k)$: the encryption of message m with the key k.
$\mathrm{D}(C,k)$: the decryption of the cipher text C with the key k.
$\mathrm{E}(m,k,IV)$: the encryption of the message m with the key k and the initial vector IV using the DES algorithm in CBC mode.
$\mathrm{D}(C,k,IV)$: the decryption of C with the key k and the initial vector IV using the DES algorithm in CBC mode.
$\mathrm{MD5}(m)$: calculates the hash value of the message m using the MD5 algorithm.
$x \| y$: the concatenation of x and y.

3.1 Network model

This paper uses the same network model described by Elsemary and Azim [7]. This network model is composed of a base station and a sink node as well as a set of randomly distributed wireless sensor nodes. It is required that every authorized deployed sensor in the network field must have a unique identification number (ID). The base station can be inserted in any suitable place whether in the field or somewhere else. It is usually connected to the sink node through a wired or wireless link. The underlying network model is depicted in Figure 1.

Figure 1 presents a network model in which the sink node is a wireless sensor node with high capabilities in terms of memory, processing, power and wireless coverage. The

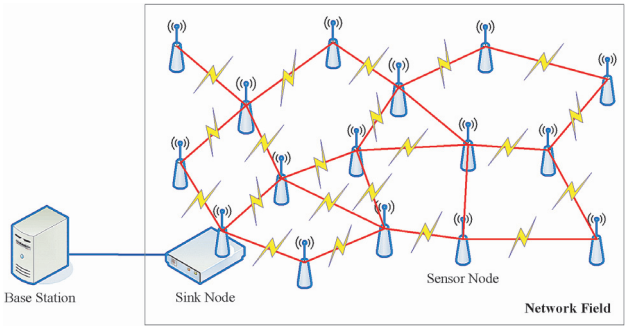


Figure 1: Network model.

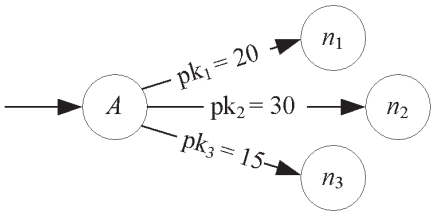


Figure 2: A network part that has a node with three parents.

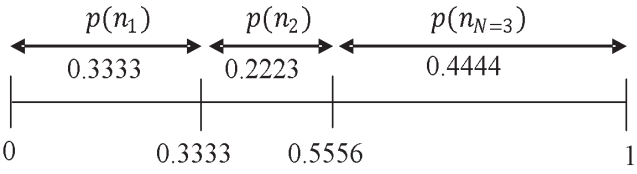


Figure 3: The selection probability line of the three parents of the node A.

sink node works as an intermediate node between the base station and the other sensors. It receives commands from the base station and then conveys them to the deployed sensors. In addition, it collects data from sensors and sends them to the base station. The other sensor nodes, which have limited capabilities in their battery-powers, memory and processing, are distributed all over the area of interest in such a way that any deployed node has at least one path to the sink node. As shown in Figure 1, the line between any two nodes means that these nodes are within the transition range of each other.

3.2 Roulette-Wheel Routing Protocol

This section introduces a novel Roulette-Wheel Routing Protocol (RWRP) that is forwarding data packets to a next hop towards the sink node during the routing process. The significance of this protocol is that routing decision of a forwarding node is not affected by any other nodes (i.e. it cannot be deceived by any other node) and network load is distributed evenly among all deployed sensors to prolong the network lifetime. The RWRP considers the number of packets sent by the forwarding node to each neighbor as forwarding metric. It implements the roulette-wheel selection algorithm to select the next forwarding node. In roulette-wheel selection algorithm, an individual is given probability of being selected that is directly proportional to its measured metric. The individual is then chosen randomly based on its probability. In this protocol, the individuals are the set of parents or siblings nodes of the forwarding node. To demonstrate the RWRP protocol, suppose a node A is the forwarding node and it needs to forward a packet through one of its parents: n_1, n_2 or n_3. Also, the current state of the node A is that it already sent 20, 30 and 15 packets to its parents: n_1, n_2 and n_3, respectively. This is depicted in Figure 2.

To choose the next node by the node A, the protocol starts calculating the total number of packets sent by the node A to its parents. This is referred to as tpk_A. For the network shown in Figure 2, tpk_A is equal to 65 which is sum of (20, 30 and 15) that represents the number of packets sent from the node A to its parent nodes n_1, n_2, n_3, respectively. This is described by the following equation:

$$tpk_A = \sum_{i=1}^{N=3} pk_i, \tag{1}$$

where pk_i is the number of packets sent from node A to its parent node n_i, and N is the number of parents of the node A which, in this case, N equals 3.

Second, the protocol assigns a weight w_i to each parent node n_i of the node A such that the parent node that has received the minimum number of packets from the node A will have the maximum weight. The weight w_i can be simply achieved as the total number of packets sent by the node A divided by the number of packets sent from the node A to its parent node n_i. This is described by the following equation:

$$w_i = \frac{tpk_A}{pk_i}. \tag{2}$$

Finally, the RWRP protocol calculates the selection probability $p(n_i)$ of selecting a parent node n_i as a next hop. The calculation of $p(n_i)$ is based on the weight w_i obtained from (2) such that the parent node that received the minimum number of packets from the node A will have the maximum probability. This can be expressed by (3) in which $p(n_i)$ equals the weight of n_i divided by the total weights of n_1, n_2 and n_3:

$$p(n_i) = \frac{w_i}{\sum_{j=1}^{N=3} w_j}. \tag{3}$$

By substituting (1) and (2) into (3), we get the selection probability of n_i as described by the following equation:

$$p(n_i) = \frac{\sum_{j=1}^{N=3} pk_j}{pk_i} * \frac{1}{\sum_{k=1}^{N=3} \frac{\sum_{j=1}^{N=3} pk_j}{pk_k}}. \tag{4}$$

The total selection probabilities of all parents should be one (i.e. $p(n_1) + p(n_2) + p(n_3) = 1$). After obtaining the selection probability of each parent, a selection probability line is achieved by summing up the selection probabilities of all parents as shown in Figure 3.

To select a parent, a random number between 0 and 1 is generated. According to the underlying example, if the number is less than or equal to 0.3333, the parent n_1 is chosen. If the number is greater than 0.3333 and less than or equal to 0.5556, the second parent n_2 is chosen. Otherwise, the parent n_3 is chosen. The calculation of the selection probability of each parent is achieved step-by-step using

Parents of node A	pk_i	tpk_A	w_i	$\sum_{i=1}^{N=3} w_i$	$P(n_i)$
n_1	20		3.250		0.3333
n_2	30	65	2.167	9.750	0.2223
n_3	15		4.333		0.4444

Table 1: Calculation of selection probabilities of the parents of the node A.

equations (1), (2) and (3) as depicted in Table 1. The RWRP protocol works in the same way on the siblings of the node A in case it has no parents. The data in Table 1 shows that the parent node that received the lowest number of packets from the node A has the highest chance to be selected as the next node to forward the data packet through it. In other words, the node A distributes the packets among its parents or siblings based on the number of packets it sent to them. The distribution of packets among the parents or siblings of the source node A in such a way means that the node A is evenly distributing the energy consumption among its parents or siblings based on its local data. If every node forwards its data packets in the same way as the node A, the energy consumption of the deployed sensor nodes will be distributed all over the underlying network. Accordingly, this will prolong the network lifetime. In addition, the forwarding node cannot be deceived by any other node since the forwarding criteria is based on the local data of the forwarding node itself. Thus, the RWRP helps to protect the network nodes against most of the attacks associated with the WSNs.

3.3 Secure Routing Protocol

The Secure Routing Protocol (SRP) provides the security for data packets by executing four phases of operations: (1) node initialization, (2) routing table establishment, (3) excluding malicious nodes and (4) routing data packets. The protocol implements the security by using the Data Encryption Standard (DES) algorithm [13] and the Message Digest version five (MD5) algorithm [17]. The DES is used in Electronic Code Book (ECB) mode to provide confidentiality or in Cipher Block Chaining (CBC) mode together with μTesla protocol to provide broadcast authentication. In addition, the DES in CBC mode along with the MD5 is used to guarantee integrity. Furthermore, a time stamp is used to prevent repetition of packets. This is described in detail throughout the underlying protocol phases.

3.3.1 Node initialization

The node initialization phase deals with nodes before their deployment into the network field. This initialization phase is described by the algorithm shown in Figure 4. First, the set of key chain $K = \{k_0, k_1, \ldots, k_n\}$ used in the μTesla is generated by randomly selecting k_n and the other keys $k_i = \text{MD5}\ (k_{i+1})$, where $0 \leq i \leq n-1$ and k_0 is the

```
InitializationAlgorithm( )
1    k_n ← generate a random key of 128 bits
2    K ← {k_n }
3    FOR i = n-1 to 0
4        k_i ← MD5 (k_i+1)
5        K ← K U k_i
6    END
7    K_sink ← { }
8    k_t ← generate a random key of 56 bits
9    IV_t ← generate an initial vector of 64 bits
10   TS_sink ← generate a random time stamp
11   k_1 ← k_1 ∈ K
12   k_56 ← f_56 (k_1), k_8 ← f_8 (k_1), k_64 ← f_64 (k_1), IV_64 ← pad (k_8) XOR k_64
13   FOREACH node i ∈ network field
14       ch_i ← generate a random variable
15       k_0 ← k_0 ∈ K
16       k_s ← k_t, IV_s ← IV_t
17       k_hi ← generate a random key of 56 bits
18       IV_hi ← generate an initial vector of 64 bits
19       k_si ← generate a random key of 56 bits
20       K_sink ← K_sink U {ID_i, k_si, k_hi, IV_hi}
21       t_i ← ID_i || f(ch_i) || TS_sink
22       MAC_i ← DES (t_i, k_56, IV_64)
23   END
24   Store K_sink into the sink node
END
```

Figure 4: Node initialization phase algorithm.

commitment of K. Each k_i has 128 bits which is equal to the output length of the MD5 algorithm. The generation of K is achieved by Steps 1 through 6 in the algorithm. Only k_0 and k_1 will be used by this paper while the other keys are used when adding new nodes into the network filed which is out of the scope of this paper. Next, each deployed node i must be initialized with four keys and two initial vectors, challenge value (*ch*), and a message authentication code (*MAC*).

The four secret keys are denoted by k_s, k_{is}, k_{hi} and k_0. The length of the first three keys is 56 bits each which is the length of the key used by the DES algorithm. The k_s is a secret key shared only among all authorized nodes deployed into the network field while the k_{is} and k_{hi} are secret keys shared only by the authorized node i and the sink node. Also node i should be initialized with two initial vectors IV_s and IV_{hi} associated with the keys k_s and k_{hi}, respectively. The length of the initial vectors is 64 bits as required by the DES. k_0 is initialized with the commentment key that belongs to the key chain K. The initialization of the keys, initial vectors, and the challenge value is achieved by Steps 14 through 19 of the algorithm described in Figure 4.

Finally, the node i is also initialized with a MAC_i which is used to detect unauthorized nodes during the excluding malicious nodes phase. Any node that does not have a valid *MAC*, is considered as a malcious node. MAC_i is an encrypted version of the token t_i using the DES algorithm in the CBC mode with the key k_{56} and initial vector IV_{64}

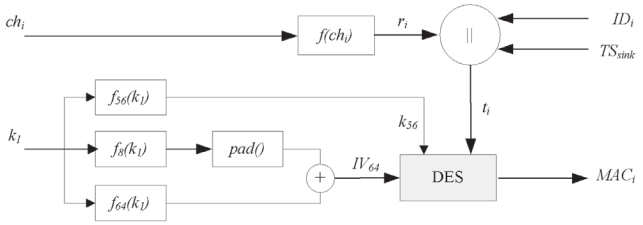


Figure 5: Generating MAC_i associated with a node i.

generated form k_1 that belongs to the key chain as depicted in Figure 5. The token t_i is the concatenation of ID_i, r_i and TS_{sink}, where ID_i is a 16 bits that provides a unique identification number for node i, r_i is the response of a challenge-response function $f(ch_i)$ corresponding to the challenge ch_i of the node i. The $f()$ is a function known by all deployed nodes. The TS_{sink} is a unique time stamp used to prevent a token reply.

Since MAC_i is authenticated by k_1 which is 128 bits using DES algorithm which has a key of length 56 bits. A conversion is done using four simple functions: $f_{56}()$, $f_8()$, $f_{64}()$ and pad(). The functions $f_{56}()$, $f_8()$ and $f_{64}()$ just extract the first 56 bits, next 8 bits, and last 64 bits of the k_1, respectively. The 56 bits is used as a key to the DES. The function pad() is used to pad the 8 bits produced by the function $f_8()$ with 56 bits of 0s. Then, the output of the functions $f_{64}()$ and pad() is XORed to produce the initialization vector IV_{64}. This assures that the MAC_i is a function in both tokens: t_i and k_1. The generation of the MAC_i is achieved through steps 11, 12, 21 and 22 of the underlying algorithm. At the end of the algorithm, the k_{sink}, which contains the secret keys and initial vectors shared with each sensor, is stored into the sink node.

3.3.2 Routing table establishment

Nodes in the routing table are classified into three categories: (1) *parent node*, (2) *sibling node* and (3) *child node*. A parent node is a node in the transmission range of another sending node and having a hop count one less than the sending node. A sibling node is a node in the transmission range of another sending node and having the same hop count as the sending node. A child node is a node in the transmission range of another sending node and having a hop count one more than the sending node. After deploying sensor nodes into the network field, the routing table establishment phase is initiated by the base station which commands the sink node to start building up routing tables of all deployed sensors. The sink node, in turn, responds by broadcasting a setup packet to all nodes within its transmission range.

The setup packet created by a node i is constructed as "*setup*.MAC_i.C_i," where the keyword *setup* is used to denote that this is a setup packet, MAC_i is the message

RoutingTableEstablishmentAlgorithm (*setUpPacket*$_i$)
1 $MAC_i \leftarrow$ extratMAC (*setUpPacket*$_i$)
2 $C_i \leftarrow$ extratMAC (*setUpPacket*$_i$)
3 $M_i \leftarrow$ D (C_i, k_s)
4 $ID_i \leftarrow$ extractID(M_i)
5 $ch_i \leftarrow$ extractCH(M_i)
6 $x_i \leftarrow$ extractX (M_i)
7 $y_i \leftarrow$ extractY (M_i)
8 $h_i \leftarrow$ extractH (M_i)
9 $TS_i \leftarrow$ extractTS (M_i)
10 $x \leftarrow x_i - x_j$
11 $y \leftarrow y_i - y_j$
12 $dist_{ij} \leftarrow$ sqrt (x*x + y*y)
13 IF $dist_{ij} \leq tr_j$ and TS_i is matches
14 IF h_j is not set before
15 Set $h_j \leftarrow h_j + 1$
16 Record the node i and its parameters into the parent list of node j
17 *setUpPacket*$_j$ $\leftarrow$ Create a new setup packet for node j
18 Transmit *setUpPacket*$_j$
19 ELSEIF $h_j > h_i$
20 Record the node i and its parameters into the parent list of node j
21 ELSEIF $h_j < h_i$
22 Record the node i and its parameters into the child list of node j
23 ELSE
24 Record the node i and its parameters into the sibling list of node j
25 END
26 END

Figure 6: Routing table establishment algorithm.

authentication code generated during the initialization phase for the node i, and C_i is the encryption of the message M_i with the shared key k_s using the DES (i.e. $C_i = \mathrm{E}(M_i, k_s)$). The message M_i is the concatenation of ID_i, ch_i, x_i, y_i, h_i and TS_i, where ID_i and ch_i are the identification number and challenge value of the node i, respectively. The parameters x_i and y_i denote the coordinates of the node i in x- and y-directions, respectively. The parameter h_i refers to the number of hop count between the node i and the sink node. Therefore, the setup packet broadcasted by sink node sets the value of h_i to 0. Finally, TS_i is a unique time stamp of the node i. Based on the setup packet format, the sink node creates its setup packet and then broadcasts it to all sensor nodes within its transmission range.

Each sensor node j receiving the setup packet executes the algorithm depicted in Figure 6. The algorithm first extracts the values of MAC_i and C_i from the packet as depicted in steps 1 and 2, respectively. Next, the node j decrypts C_i with the key k_s using the DES algorithm to obtain the message M_i (i.e. $M_i = \mathrm{D}(C_i, k_s)$) as shown in Step 3. Then, the parameters ID_i, ch_i, x_i, y_i, h_i and TS_i that make up the message M_i are extracted by steps 4 through 9. For example, the function *extracted* (M_i) in step 4 extracts the value of ID_i from the message M_i. Next, node j calculates distance denoted by $dist_{ij}$ between itself and the node i based on its values of x_j and y_j and the received x_i and y_i as shown in steps 10, 11 and 12.

Next, the node j checks both its $dist_{ij}$ and the time stamp TS_i received from the node i. If the $dist_{ij}$ is out of the transmission range of the node j denoted by tr_j or TS_i is not matched (step 13), the node j simply drops the packet. Otherwise, it checks its hop count h_j. If h_j is

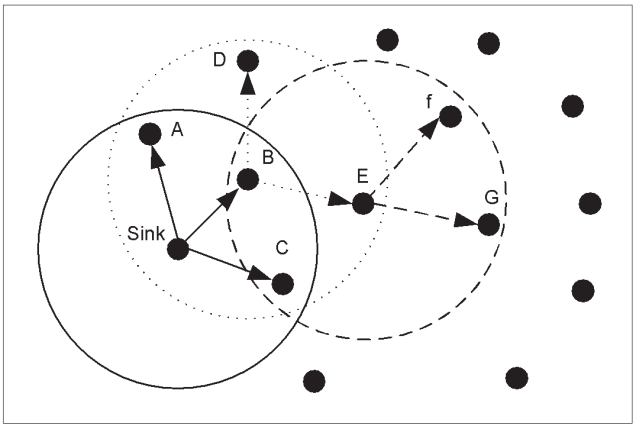


Figure 7: Setup packets are flooded during building up routing tables.

not set before, it increments h_j, records the node i and its associated parameters into its parent list. Next, the node j creates its setup packet *setUpPacket*$_j$ and then broadcasts it to all nodes within its transmission range (steps 14 through 18). Otherwise, it checks that h_j is greater than h_i. If so, it records the node i and its associated parameters into its parent list (steps 19 and 20). Otherwise, the node j tests if h_j is less than h_i. If so, it records the node i and its associated parameters into its child list (steps 21 and 22). Otherwise, it records the node i and its associated parameters into its sibling list (steps 23 and 24).

To illustrate the process of building the routing tables, a network is constructed in Figure 7. Upon the sink it is commanded from the base station to build the routing tables, it constructs its setup packet with hop count of zero (i.e. $h_{sink} = 0$) and broadcasts it to all nodes in its transmission range. In this case, the nodes A, B and C in the solid circle will receive the setup packet. Since the hop counts of these nodes have not been set before, each of the nodes A, B and C marks the sink node as its parent and sets its hop-count with the value of 1.

Next, every node with hop-count of l, in this case A, B and C, constructs its own setup packet and broadcasts it to all nodes within its transmission range. For example, suppose that node B in the dotted circle sends this setup packet which in turn will be received by its neighbor nodes: sink, A, C, D and E. Each of these nodes decrypts the setup packet to extract the included parameters and accordingly updates its routing table. The sink node marks the node B as a child node since the value of h_{sink} is less than the value of h_B. The nodes A and C mark node B as a sibling node because each of the values of h_A and h_C is equal to the value of h_B. Finally, each of the nodes D and E sets its hop count to 2 (i.e. $h_B + 1$) and marks the node B as its parent since neither the hop count h_D nor the hop count h_E is set before. Thus, each of these nodes creates its setup packet and broadcasts it to all nodes in its transmission range. This broadcasting process is repeated hop-by-hop until it establishes the remaining routing tables.

A routing table of a sensor node i has information about each sensor j that is in its transmission range. This information includes ID_j, ch_j, x_j, y_j, TS_j, MAC_j, $class_j$ *and* pk_j. The $class_j$ is set based on the classification of the node j as parent, sibling or child. The pk_j is used to keep track of the number of packets sent from the node i to the node j. Furthermore, the routing table includes an *active* field which is set to either false or true to denote whether the node j is operational or not. Operational means that the node can receive and forward a data packets. Up to this phase, all deployed nodes are not operational (i.e. *active* is set to false) until excluding all malicious nodes that may be found in the routing tables.

3.3.3 Excluding malicious nodes

This phase is mainly used to exclude any malicious nodes that may be found in the routing tables. It starts when the sink node broadcasts the authentication message *AM*, where *AM* is the encryption of the concatenation of ID_{sink}, k_1 and TS_{sink} using the shared key k_s along with its associated initial vector IV_s (i.e. $AM = \mathrm{E}((ID_{sink} \| k_1 \| TS_{sink}), k_s, IV_s)$, where ID_{sink} is the identification number of sink node, k_1 belongs to the key chain generated during the initialization phase and TS_{sink} is a new time stamp.

Each authorized node i receives the authentication message *AM*, it first decrypts the *AM* to obtain the plaintext m (i.e. $m = \mathrm{D}(AM, k_s, IV_s)$. Next, it extracts the parameters: ID_{sink}, k_1 and TS_{sink} from m. Then, the node examines the time stamp TS_{sink} to check whether the received *AM* is a reply message or not. If so, it simply ignores the message. Otherwise, it authenticates k_1 by performing $k = \mathrm{MD5}(k_1)$. If k is not equal to the k_0 stored during the initialization phase, the node i simply ignores the message. Otherwise, the node i obtains the key k_{56} and the initialization vector IV_{64} from the key k_1 as discussed in the last paragraph of the initialization phase.

Using k_{56} and IV_{64}, the sensor node i decrypts MAC_j associated with each sensor node j belonging to its routing table to reveal the token t_j (i.e. $t_j = D$ (MAC_j, k_{56}, IV_{64})), where t_j was constructed from the concatenation of ID_j, r_j and TS_j during the routing phase which are easy to be obtained. These values are compared against the data received from the sensor node j during the routing phase. This data includes the challenge ch_j and time stamp TS_j of the sensor node j. If both are matched, then the sensor node j is considered as an authorized node. Otherwise, it is removed from the routing table of the sensor node i. Also, the *active* field associated with the node i is set to true (i.e. the sensor node i becomes operational). Therefore, after

SourceNodeAlgorithm ( )
1 $d_i \leftarrow$ sensed data form node i
2 $pk_{no} \leftarrow$ generate a packet number for node i
3 $TS_i \leftarrow$ generate a time stamp for node i
4 $s_i \leftarrow$ E (MD5 ($ID_{sink} \| ID_i \| d_i \| pk_{no} \| TS_i$), k_{hi}, IV_{hi})
5 $b_i \leftarrow$ E (($d_i \| pk_{no} \| TS_i$), k_{is})
6 $pk_{is} \leftarrow$ ($ID_i \| b_i \| s_i$)
7 Select next node j from routing table of i
8 $nTS_i \leftarrow$ generate a time stamp for node i
9 $packet_{ij} \leftarrow$ $E((ID_i \| pk_{is} \| nTS_i), k_s, IV_s)$
10 Transmit $packet_{ij}$ to node j
END

Figure 8: Data routing algorithm performed by a source node i.

this phase, the network goes into an operational state which means that each deployed node can receive and forward data packets.

3.3.4 Routing data packets

This final phase is concerned with routing data packets in a secure and energy-efficient manner. A data packet is intended to be delivered from a source node to the sink node through a set of intermediate nodes. The data packet is processed by all nodes that it passes through during its transmission from a source to the destination (sink). The nodes along the route from the source to the sink node can be classified based on their functionality into three categories: (1) source node, (2) intermediate nodes and (3) sink node. The operation achieved by each category is presented in the rest of this section.

A sensor node i from the first category performs the algorithm described in Figure 8 when it has a new data ready to be sent to the sink node. The algorithm has two parts: the first part (steps 1 to 6) and the second part (steps 7 to 10). The first part provides confidentiality, integrity, authentication, and non-repetition for data being sent from a source node to the sink node while the second part achieves the authentication, and non-repetition for a data packet between sending and receiving nodes in the transmission range of each other.

The first part of the algorithm starts by sensing the data d_i of the node i and then generates both a packet number pk_{no} and a new time stamp TS_i. This is described by the underlying algorithm in steps 1, 2 and 3. Next, it calculates the signature s_i of the d_i to achieve the integrity and authentication as well as non-repetition of d_i. The s_i is produced as a function of the identifications of the sink node (ID_{sink}) and node i (ID_i), d_i, pk_{no}, TS_i and the key k_{hi} shared only between the sink and the sensor node i together with its associated initial vector IV_{hi} (see step 4 in Figure 8). The ID_i and ID_{sink} are used to indicate that it is created by the node i and intended to the sink node. The d_i is used to indicate that the signature is produced for this specific data while

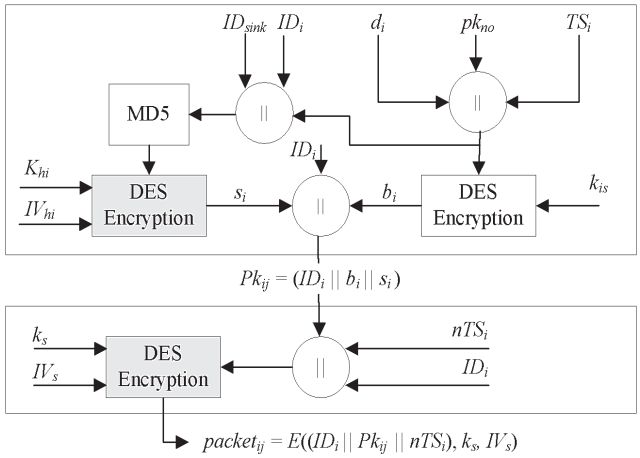


Figure 9: Generating a packet $packet_{ij}$ forwarded to the node j by the source node i.

pk_{no} is a new packet number. The TS_i is used to prevent the non-repetition of the data. The k_{hi} is used to provide authentication since it is shared only by the sink and the source node i.

Finally, the confidentiality of d_i together with pk_{no} and TS_i is provided by encrypting them with the key k_{is} shared only between the sink and the node i. The output of the encryption is denoted by b_i as shown in step 5. The intended data pk_{is} from the node i to the sink node is composed of the concatenation of ID_i, b_i and s_i (step 6).

The second part of the algorithm starts by selecting a next node j from the routing table of the node i to forward the pk_{is} through it (step 7). The selection of the sensor node j is based on the novel approach introduced in Section 3.2. Next, the algorithm generates a new time stamp $n_1 TS_i$ (step 8) to prevent non-repetition of the packet $packet_{ij}$ that will be sent form the node i to the node j. The $packet_{ij}$ is produced as the encryption of the concatenation of ID_i, pk_{is} and nTS_i using the key k_s and its associated initial vector IV_s that is shared by all authorized nodes (step 9). Finally, the packet $packet_{ij}$ is transmitted to the node j (step 10). The first and second parts of the algorithm are also depicted in more detail in top and down boxes of Figure 9, respectively.

When the node j belonging to the second category receives the packet $packet_{ij}$ sent by the sensor node i, it executes the algorithm steps shown in Figure 10. The algorithm first decrypts the packet $packet_{ij}$ with k_s and IV_s to reveal the value of ID_j, pk_{ij} and nTS_j as indicated by steps 1 to 4. Next, it checks if the ID_i belongs to its routing table to assure that the $packet_{ij}$ is encrypted by an authorized node and then tests the nTS_i to verify whether the $packet_{ij}$ is a reply packet or not. If so, it is simply dropped (steps 5 and 6). Next, the sensor node j selects a next node w from its routing table as before and then generates a unique time stamp nTS_j (steps 7 and 8). Finally,

```
RecievedPacketByIntermediateNode (packet_ij)
1       m ← D(packet_ij, k_s, IV_s)
2       ID_i ← extractID(m)
3       pk_is ← extractPK(m)
4       nTS_i ← extractTS(m)
5       IF ID_i ∈ routing table of node j
6           IF nTS_i matches time stamp of node i
7               Select a next node w from routing table of j
8               nTS_j ← generate a unique time stamp
9               Packet_jw ← E((ID_j || pk_is || nTS_j), k_s, IV_s)
10              Transmit packet_jw to the node w
11          END
12      END
END
```

Figure 10: Data routing algorithm performed by an intermediate node.

```
SinkNodeAlgorithm (packet_ij)
1       m ← D(packet_ij, k_s, IV_s)
2       ID_j ← extractID(m)
3       pk_is ← extractPK(m)
4       nTS_j ← extractTS(m)
5       IF ID_j ∈ routing table of sink node
6           IF nTS_j matches time stamp of node j
7               ID_i ← getID(pk_is)
8               b_i ← getB(pk_is)
9               s_i ← getS(pk_is)
10              temp ← D (b_i, k_is)
11              d_i ← getD (temp)
12              pk_no ← getPkno (temp)
13              TS_i ← getTS (temp)
14              IF TS_i matches time stamp of node i
15                  s ← E (MD5 (ID_sink || ID_i || d_i || pk_no || TS_i), k_hi, IV_hi)
16                  IF s_i = s
17                      Use sensed data d_i
18                  ELSE
19                      Ignore d_i
20                  END
21              END
22          END
23      END
END
```

Figure 11: Data routing algorithm performed by the sink node.

it creates the packet $packet_{jw}$ and sends it to the node w as shown by steps 9 and 10. This process is repeated by all intermediate nodes (i.e. a node of the second category) included in the path from the source to the sink node until the packet $packet_{jw}$ reaches the sink node.

Once the sink node which represents the 3rd category receives the packet $packet_{jw}$ (where in this case, w and j refer to the sink node and a node of its children, resp.), it performs the algorithm steps shown in Figure 11. Like the algorithm (Figure 10) performed by an intermediate node, the underlying algorithm starts by executing the first six steps to recover the ID_j, pk_{is} and nTS_j and tests identity of ID_j, as well as matches the time stamp nTS_j against the time of the node j. If both ID_j and nTS_j are valid, it means that $packet_{jw}$ is a new and authentic packet. Therefore, the algorithm continues its operation by extracting the values of ID_{sink}, ID_i, d_i, pk_{no}, TS_i from the pk_{is} created by the source node i as introduced in Figure 8. This is achieved through executing steps 7 to 13. For example, step 9: $s_i \leftarrow \text{getS}(pk_{is})$ is used to extract the signature s_i from the pk_{is}. If the recovered TS_i is a valid time stamp from the node i, it means that d_i is a new data and not a reply one.

Next, the algorithm checks the integrity of d_i by calculating the signature s from the recovered parameters as indicated by Step 15 and then compared with the received signature s_i that is created by the source node i performing the algorithm shown in Figure 8. Specifically, it is achieved by step 4: $s_i \leftarrow \text{E}(\text{MD5}(ID_{sink}\|ID_i\|d_i\|pk_{no}\|TS_i), k_{hi}, IV_{hi})$. If s and s_i are equal, it assures that nothing is tempered with the data d_i and it is processed by the sink node. Otherwise, d_i is ignored by the sink node. This is depicted by steps 16 to 20 of the underlying algorithm. Since the signature s_i is produced as the encryption of the output of the *MD5* with the key k_{hi} and the initial vector IV_{hi} shared only by the node i and the sink node, it cannot be produced by any other node except the node i or the sink node. Therefore, the sink node believes that the s_i and d_i are sent from the node i.

3.4 Security aspects of EESRP

This section discusses the security features of EESRP protocol. The EESRP protocol provides protection for WSNs against most of attacks on both application data and routing protocols. The protection against application data attacks, such as revealing, tampering, repeating, spoofing of data, is achieved by providing encryption, digital signature and freshness features for conveyed data between a source and the sink node. Also the EESRP provides security feature to guard in particular against attacks on routing protocols that draw traffic by advertising high quality path to the sink node. This security feature of EESRP comes from the distinctive way used to construct the path between the source and the sink node. In addition, each node is permitted to only receive from and send to authentic nodes.

The path is constructed randomly from the source to the sink. In other words, each node along the route starting from the source randomly selects a next node towards the sink form its routing table. The routing table of each node contains only its authentic neighbors. Therefore, it is extremely hard to a malicious node to include itself on a path during its establishment. Karlof and Wagner [11] summarized attacks on routing protocols and their countermeasures. These attacks include tampering routing information, sinkhole attacks, selective forwarding attacks, wormhole attacks and spoofing attacks. The rest of this section shows how the EESRP protocol defends against these attacks.

Tampering routing information attack is defended by the EESRP protocol. After the execution of the *excluding malicious nodes* phase, the behavior of the EESRP protocol does not allow any node to update its routing table based on advertised information. Consequently, a malicious node cannot tamper with the routing tables of other nodes.

In selective forwarding attacks, malicious nodes drop part or all received packets so that they are not propagated any further. These types of attacks are typically most effective when the attacker is explicitly incorporated into the routing path during the data flow. The EESRP protocol prevents a malicious node to be included on the routing path. A forwarding node randomly selects a next node towards the sink from its routing table that only includes authentic nodes.

Next, sinkhole attacks try to entice traffic from sensor nodes to the sink node. These types of attacks are hard to be defended in protocols that utilize advertised information such as Min-Hop and PEW protocols because this information is not easy to be verified. However, in EESRP protocol, the construction of the routing path does not depend on advertised information. Consequently, an attacker employing a sinkhole attack cannot deceive a forwarding node to send its packets to a malicious node launched by this attack.

Next, wormhole attacks use a private and out-of-band channel between at least two malicious nodes to forward data packets from one place to another in the network through this channel. One malicious node eavesdrops data packets from its place and sends them through the established channel to the other malicious node which in turn forwards them to a node of its neighborhood. This type of attack is not completed in the EESRP protocol since when a node receives a data packet, it first checks whether it comes from a valid neighbor or not. If so, it will consider the packet. Otherwise, it simply drops the packet. In this case, the receiving node will drop any packet comes through the established channel.

Finally, in spoofing attack, a malicious node impersonates a valid node. It floods its neighbors with packets that have the identity of valid IDs. Fortunately, valid data packets can only be formed by valid nodes. Thus, this attack can be launched with reply packets. In both cases, the receiving nodes will detect these kinds of packets and simply drop them.

4 Experimental results

To verify the performance of the proposed EESRP protocol, a WSN network simulator is developed. In the simulations, the performance of the proposed protocol is compared with the performance of PEW and Min-Hop routing protocols under attack-free and attack conditions.

The simulation environment uses a random network topology similar to the one used by [7] in which 300 sensor nodes were randomly scattered in a fixed area of $50 \times 50\,\mathrm{m}^2$, and the sink node is located at the lower left corner. A source node is choosing randomly to send a data packet of size 1024 bits to the sink node. The energy update packet size is assumed to be 64 bits for both PEW and Min-Hop protocols. Also the value of the maximum energy weight $W_{\max}$ is chosen to be 4 for the PEW protocol. In our simulation, a total of 4000 data packets were generated and fed to the three protocols, simultaneously. Each sensor node is initialized with $1000,000$ units of energy (i.e. $E_{\max}$) and assumed to spend 1 energy unit for receiving and 1 energy unit for transmitting one data bit. In addition, each sensor node has a maximum transmission range of 5 m and also a detection range of 5 m. During the course of our results, three experiments were carried out. The first experiment is performed under normal operation. The second experiment is executed under the sinkhole attack and the third experiment is operated under the spoofing attack.

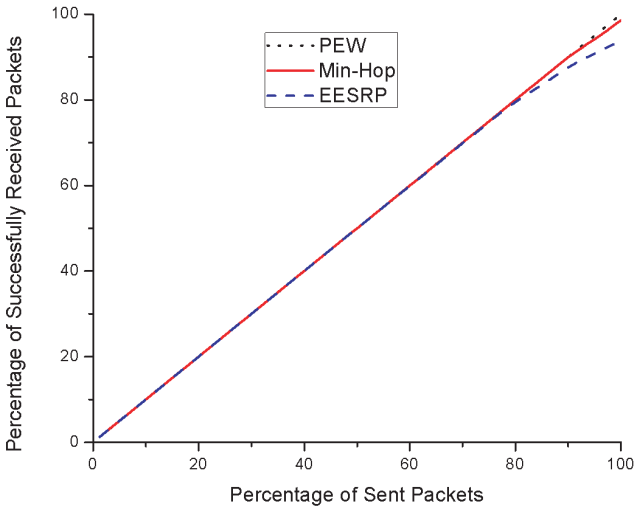


Figure 12: Percentage of successfully received packets during attack-free condition.

In the first experiment, we examine the performance of the three protocols under attack-free condition. In this experiment, the percentage of successfully received packets versus the percentage of sent packets is presented for each protocol in Figure 12.

The results shown in Figure 12 indicate that, up to 80% of the sent packets, all of the three protocols have a similar behavior. After the 80% of the sent packets, the difference between the percentage of successfully received packets of the proposed protocol and that of both the PEW and Min-Hop routing protocols did not cross over 6% at its maximum. However, this sacrifice in the percentage of successfully received packets using the EESRP is due to the overhead bits that are added to each packet as a result of encryption and the addition of a hash value to provide

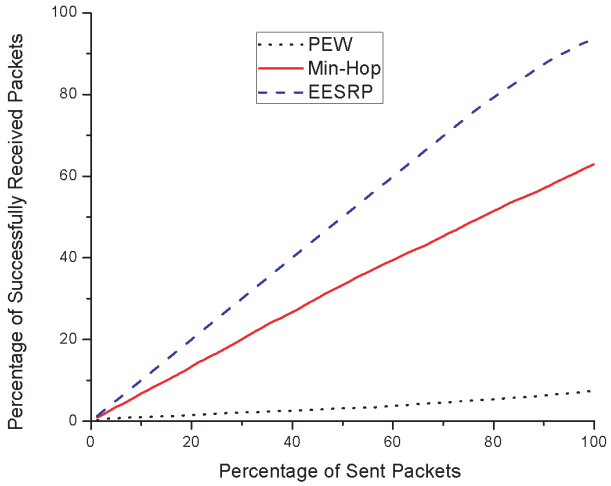


Figure 13: Percentage of successfully received packets during sinkhole attack.

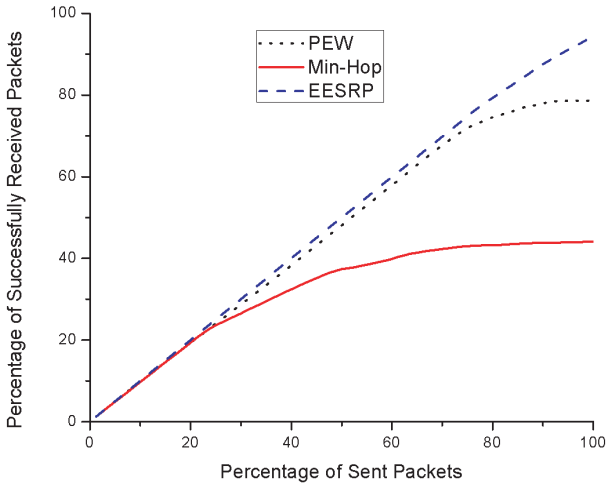


Figure 14: Percentage of successfully received packets during spoofing attack.

confidentiality, integrity, authentication and non-repetition of messages.

In the second experiment, a sinkhole attack is injected into the underlying network. Similar to the first experiment, the second experiment investigates the percentage of successfully received packets versus the percentage of sent packets for each protocol. In this experiment, the sinkhole attack is implemented as follows: three unauthorized nodes were randomly added to the network of each approach. The rule of each of these fake nodes is to deceive its neighbors to forward their packets through it. For example, a fake node misleads its neighbors in the Min-Hop by updating their routing tables with the highest energy level since the protocol allows a node to update the energy levels of its neighbors. Similarly, a fake node in the PEW approach misleads its neighbors that it has the best path by updating their routing tables with the value of its PEW parameter during the update process in which each node conveys its PEW parameters to its neighbors. In these two cases, the fake nodes lured their neighbors to forward packets through them. Consequently, the fake nodes will have ability to drop these packets. On the other hand, in the proposed EESRP approach, these nodes will be detected and removed from the routing tables of their neighbors during the *excluding malicious node* phase. Figure 13 depicts the behavior of the underlying protocols under the sinkhole attack.

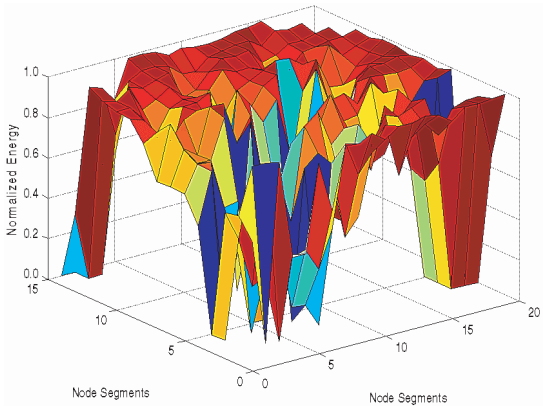
The results depicted in Figure 13 show that the performance of the proposed protocol surpasses the performance of the other two approaches under the sinkhole attack. It is also clear that the performance of the proposed EESRP protocol is not affected by the sinkhole attack while the performance of the other two protocols does. In particular, the PEW approach provides a very low percentage of successful packets due to the global calculations of the path information. This forces most of the packets to pass through the unauthorized nodes which in turn drop them. The performance of the Min-Hop protocol is moderate between the performance of the PEW and EESRP protocols.

During the last experiment, a spoofing attack is implemented to investigate its effect on the deployed network in each protocol. The spoofing attack can be easily implemented by inserting one or more unauthorized nodes into the field. Actually, three unauthorized nodes are inserted; each around a corner of the three corners of the field other than the corner that contains the sink node. The rule of these unauthorized nodes is to continuously send malicious packets each of size 1024 bits (similar to the size of the data packet) during the network operation. The percentage of successfully received packets is presented in Figure 14.

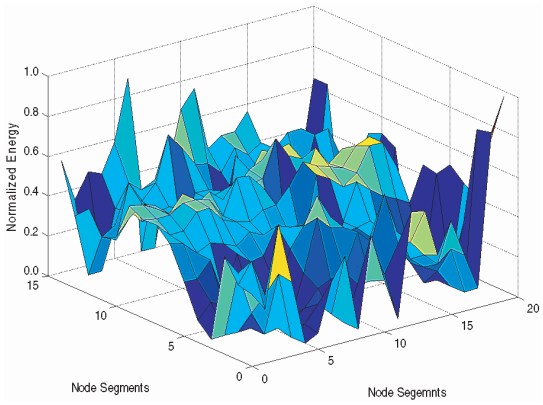
The results depicted in Figure 14 clearly show that the performance of the proposed protocol is better than the performance of the other two protocols. It is also clear that the EESRP approach detected the malicious nodes and protected the network against them. On the other hand, these malicious nodes affect the Min-Hop and PEW approaches by consuming a lot of energy which shorten the lifetime of the associated networks, especially in a network deploying the Min-Hop protocol.

To further investigate the performance of the proposed protocol, the energy distribution of each network under the spoofing attack is visualized in Figure 15. To achieve this, the network area of $50 \times 50\,\mathrm{m}^2$ is segmented into 20×15 segments (i.e. 300 segments) such that nearly each sensor node fits into a segment. The normalized energy distribution depicted in Figure 15 is considered when the percentage of sent packets reaches 75% of the total number of packets.

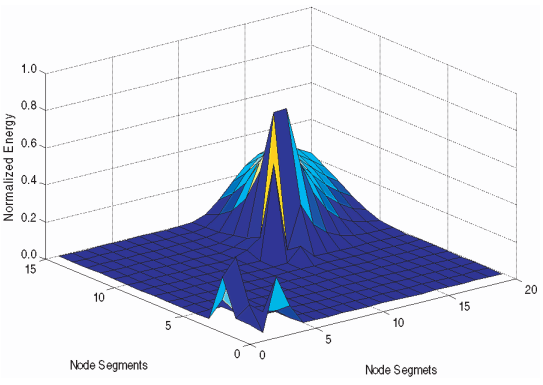
Figure 15(a) shows the energy distribution of the network deploying the EESRP and it clearly shows that each malicious node only consumed the energy of the nodes in its neighborhood due to receiving malicious packets. After



(a) EESRP



(b) PEW



(c) Min-Hop

Figure 15: The energy distribution of the three networks under the spoofing attack.

receiving malicious packets, the neighbors detected these packets as malicious and thus they simply drop them. Therefore, the other nodes in the network will not be affected.

In the same way, Figure 15(b) depicts the energy distribution of the network deploying the PEW protocol. The malicious packets sent by the malicious nodes were conveyed throughout the network. Therefore, this shortens its lifetime due to the energy consumption of sensors that receive and forward these packets.

Finally, Figure 15(c) depicts the energy distribution of the network deploying the Min-Hop protocol. The results in Figure 15(c) also indicate that most of the deployed nodes were died due to the depletion of their batteries as a result of continuous receive and forward processes of the fake packets.

5 Conclusions

This paper introduced an Energy-Efficient Secure Routing Protocol (EESRP) that provides security for data packets during its way from a source node to the sink node and also prolongs the network lifetime by evenly distributing network load among deployed sensors. The EESRP was achieved by developing two individual protocols: Roulette-Wheel Routing Protocol (RWRP) and Secure Routing Protocol (SRP). The RWRP is concerned with routing a data packet while SRP is interested in securing it during its traveling from a source to the sink. The RWRP employed roulette-wheel selection algorithm to select a next hop in the forwarding process. The next hop selection is based on the forwarding node information. Therefore, it cannot be deceived by any other node. The SRP implemented μTesla protocol together with the MD5 and DES algorithms to provide broadcast authentication, confidentiality, integrity, authentication and non-repetition of conveyed packets. The μTesla was used to achieve broadcast authentication using symmetric encryption algorithms instead of asymmetric encryption algorithms due to the fact that the symmetric algorithms are faster than the asymmetric counterpart. The performance of the EESRP was compared to the PEW and Min-Hop protocols under various conditions: attack-free, sinkhole attack and spoofing or Sybil attack. The results showed its promising efficiency and security. However, this protocol is applied only to networks with stationary sensor nodes. Therefore, the future work of this paper is to extend the proposed protocol to be applied by networks with mobile sensor nodes.

Acknowledgment This work is supported by the *Deanship* of Scientific *Research*, *Taibah University*, KSA.

References

[1] K. Akkaya and M. Younis, *A survey on routing protocols for wireless sensor networks*, Ad Hoc Networks, 3 (2005), 325–349.

[2] J. N. Al-Karaki and A. E. Kamal, *Routing techniques in wireless sensor networks: a survey*, IEEE Wireless Communications, 11 (2004), 6–28.

[3] M. M. A. Azim, *MAP: a balanced energy consumption routing protocol for wireless sensor networks*, Journal of Information Processing Systems, 6 (2010), 295–306.

[4] T. Bayrem, R. Slim, and B. Noureddine, *A novel secure and multipath routing algorithm in wireless sensor networks*, in Proceedings of the 2010 International Conference on Data Communication Networking, Athens, Greece, 2010, 25–34.

[5] S.-S. Chiang, C.-H. Huang, and K.-C. Chang, *A minimum hop routing protocol for home security systems using wireless sensor networks*, IEEE Transactions on Consumer Electronics, 53 (2007), 1483–1489.

[6] H. Deng, X. Sun, B. Wang, and Y. Cao, *Selective forwarding attack detection using watermark in WSNs*, in IEEE ISECS International Colloquium on Computing, Communication, Control, and Management (ICCCM 2009), Sanya, August 2009.

[7] A. M. El-semary and M. M. A. Azim, *Path energy weight: a global energy-aware routing protocol for wireless sensor networks*, in Proceedings of IEEE Conference on IFIP Wireless Days 2010, Venice, Italy, October 2010.

[8] W. R. Heizelman, A. Chandrakasan, and H. Balakrishnan, *Energy-efficient communication protocol for wireless micro sensor networks*, in IEEE Proceedings of the Hawaii International Conference on System Sciences (HICSS), Washington DC, January 2000, 3005–3024.

[9] C. Intagagonwiwat, R. Govindan, and D. Estrin, *Directed diffusion: a salable and robust communication paradigm for sensor networks*, in Proceedings of the 6th ACM International Conference on Mobile Computing and Networks (Mobicom'00), Boston, MA, August 2000, 56–67.

[10] M. R. Kanjee, K. Divi, and H. Liu, *A physiological authentication scheme in secure healthcare sensor networks*, in Proccedings of 7th Annual IEEE Communications Society Conference on Mesh and Ad Hoc Communications and Networks (SECON), Boston, MA, June 2010.

[11] C. Karlof and D. Wagner, *Secure routing in wireless sensor networks: attacks and countermeasures*, Ad Hoc Networks, 1 (2003), 293–315.

[12] S. Lindsey and C. S. Raghayendra, *PEGASIS: Power-efficient gathering in sensor information system*, IEEE Aerospace Conference Proceedings, 3 (2002), 1125–1130.

[13] National Institute of Standards and Technology, *FIPS-46-3: Data Encryption Standard (DES)*. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, 1999.

[14] P. Ning and D. Liu, *Broadcast authentication and key management for secure sensor networks*, in Handbook of Sensor Networks: Algorithms and Architectures, I. Stojmenović, ed., John Wiley & Sons, Chichester, 2005.

[15] A. Perrig, R. Szeczyk, J. D. Tygar, V. Wen, and D. E. Culler, *SPINS: security protocols for sensor networks*, Wireless Networks, 8 (2002), 521–534.

[16] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, *SPINS: security protocols for sensor networks*, in Proceedings of 7th Annual International Conference on Mobile Computing and Networking (MobiCom'01), Rome, Italy, 2001.

[17] R. L. Rivest, *RFC 1321: The MD5 Message-Digest Algorithm*. MIT Laboratory for Computer Science and RSA Data Security, Inc., 1992.

[18] M. Younis, N. Krajewski, and O. Farrag, *Adaptive security provision for increased energy efficiency in wireless sensor networks*, in IEEE 34th Conference on Local Computer Networks (LCN 2009), Zurich, Switzerland, October 2009.

[19] Y. Yu, R. Govindan, and D. Estrin, *Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks*, Tech. Report UCLA-CSD TR-01-0023, University of California at Los Angeles Computer Science Department, May 2001.

[20] Y. Zhang, J. Yang, W. Li, L. Wang, and L. Jin, *An authentication scheme for locating compromised sensor nodes in WSNs*, Journal of Networks and Computer Applications, 33 (2010), 50–60.

[21] J. Zhou, C. Li, Q. Cao, and Y. Shen, *An intrusion-tolerant secure routing protocol with key exchange for wireless sensor networks*, in Proceeding of the 2008 IEEE International Conference on Information and Automation, Changsha, China, June 2008.